

The Topology of Asynchronous Byzantine Colorless Tasks

Hammurabi Mendes^a, Christine Tasson^b, and Maurice Herlihy^{*a}

^a Computer Science Dept., Brown University, Providence, RI, USA,
`{hmendes,mph}@cs.brown.edu`

^b Univ. Paris Diderot, Sorbonne Paris Cité, PPS, UMR 7126, CNRS,
 F-75205, Paris, France,
`Christine.Tasson@pps.univ-paris-diderot.fr`

Abstract

In this paper, we extend the topological model that characterizes task solvability in crash-failure systems to colorless tasks in Byzantine asynchronous systems. We give the first theorem with necessary and sufficient conditions to solve *arbitrary* colorless tasks in such model, capturing the relation between the total number of processes, the number of faulty processes, and the topological structure of the task's simplicial complexes. In the interim, we provide novel asynchronous protocols for k -set agreement and barycentric agreement with optimal Byzantine fault tolerance.

Regular submission.

^{*}Supported by NSF 000830491.

1 Introduction

A *task* is a distributed coordination problem in which each process starts with a private input from a finite set, communicates with other processes, and eventually decides on a private output, also from a finite set.

One of the central questions in distributed computing is characterizing which tasks can be solved in which models of computation. Tools adapted from combinatorial topology have been successful in characterizing task solvability in synchronous and asynchronous *crash-failure* models [13], where faulty processes simply halt and fail silently. In this paper, we extend the approach to the asynchronous *Byzantine* model, where communication has unbounded delay and faulty processes can display arbitrary behavior.

We focus on an important class of tasks called *colorless* tasks [4], which includes well-studied problems such as consensus [8], k -set agreement [7], and approximate agreement [1]. As explained more formally below, a colorless task is one that can be defined entirely in terms of *sets* of input and output values assigned, without the need to specify which value is assigned to which process, or how many times an assigned value appears¹.

1.1 Our Contributions

Our first contribution is to **extend** the application of the topological model from [13], formerly used to characterize solvability in crash-failure systems, to colorless tasks in asynchronous Byzantine systems. This approach leads to novel conclusions, suggesting that the model is well-suited and robust among a multitude of communication and failure abstractions. Besides, it opens up new questions, such as how to further extend the application to colored tasks, as well as other timing models. Background information on the topological model in distributed computing is presented in Sec. 2.

Our second contribution is to give **new protocols** for k -set agreement and barycentric agreement in the Byzantine-failure model. It is well-known that asynchronous Byzantine consensus is impossible [8], as well as wait-free asynchronous set-agreement [4, 13, 17]. Our protocols limn the boundary between the possible and impossible in the presence of Byzantine failures. Using powerful algorithmic tools, presented in Sec. 3, our k -set agreement and barycentric agreement protocols, which are discussed in Sec. 4, will provide the appropriate support for our main theorem.

¹ The *renaming* task [2] is an example of a task that is *not* colorless, as each process chooses a distinct name.

Finally, our principal contribution is to give the **first theorem** with necessary and sufficient conditions to solve *arbitrary colorless tasks* in the asynchronous Byzantine model. The task itself is defined in terms of a pair of combinatorial structures called *simplicial complexes* [16, 15]. Whether a task is solvable is equivalent to the existence of a certain structure-preserving map between the task’s simplicial complexes. This equivalence captures the relation between $n + 1$, the number of processes, t , the number of failures, and the topological structure of the task’s simplicial complexes². While an analogous characterization has long been known for crash failures [13], our solvability theorem, presented in Sec. 5, is the first such characterization for Byzantine failures.

2 Topological Model

We now describe some basic notions from combinatorial topology, and how they are applied to model concurrent computation. The next section is just an overview; for more detail see Munkres [16] or Kozlov [15].

2.1 Combinatorial Tools

A *simplicial complex* is a finite set V along with a collection of subsets \mathcal{K} of V closed under containment. An element of V is a *vertex* of \mathcal{K} . Each set in \mathcal{K} is called a *simplex*, usually denoted by lower-case Greek letters: σ, τ . A subset of a simplex is called a *face*. The *dimension* $\dim(\sigma)$ of a simplex σ is $|\sigma| - 1$. We use “ k -simplex” as shorthand for “ k -dimensional simplex”, and similarly for “ k -face”. The dimension $\dim(\mathcal{K})$ of a complex is the maximal dimension of its simplices. The set of simplices of \mathcal{K} of dimension at most ℓ is a subcomplex of \mathcal{K} , called the ℓ -*skeleton* of \mathcal{K} , denoted $\text{skel}^\ell(\mathcal{K})$.

Let \mathcal{K} and \mathcal{L} be complexes. A *vertex map* f carries vertices of \mathcal{K} to vertices of \mathcal{L} . If f also carries simplices of \mathcal{K} to simplices of \mathcal{L} , it is called a *simplicial map*. A *carrier map* Φ from \mathcal{K} to \mathcal{L} takes each simplex $\sigma \in \mathcal{K}$ to a subcomplex $\Phi(\sigma) \subseteq \mathcal{L}$, such that for all $\sigma, \tau \in \mathcal{K}$, with $\sigma \subset \tau$, we have $\Phi(\sigma) \subset \Phi(\tau)$. A simplicial map $\phi : \mathcal{K} \rightarrow \mathcal{L}$ is *carried by the carrier map* $\Phi : \mathcal{K} \rightarrow 2^\mathcal{L}$ if, for every simplex $\sigma \in \mathcal{K}$, we have $\phi(\sigma) \in \Phi(\sigma)$.

Although we have defined simplices and complexes in a purely combinatorial way, they can also be interpreted geometrically. An n -simplex can be identified with the convex hull of $(n + 1)$ affinely-independent points in the Euclidean space of appropriate dimension. This geometric interpretation

² Our results extend to the core/survivor-set model [14] (Section 5.1).

can be extended to complexes. The point-set underlying such a *geometric complex* \mathcal{K} is called its *polyhedron*, and is denoted $|\mathcal{K}|$.

Given a simplicial map $\phi : \mathcal{K} \rightarrow \mathcal{L}$ (resp. carrier map $\Phi : \mathcal{K} \rightarrow 2^{\mathcal{L}}$), the polyhedrons of every simplex in \mathcal{K} and \mathcal{L} induce a continuous simplicial map $\Phi : |\mathcal{K}| \rightarrow |\mathcal{L}|$ (resp. carrier map $\Phi : |\mathcal{K}| \rightarrow |2^{\mathcal{L}}|$). In that case, ϕ is carried by Φ if, for every simplex $\sigma \in \mathcal{K}$, we have that $|\phi(\sigma)| \subseteq |\Phi(\sigma)|$.

Continuous maps can also induce combinatorial ones, but this requires more mathematical machinery. In this work, we need only to use the simplicial approximation theorem [16], a foundational result in combinatorial topology, leaving the details for the interested reader [16, 15].

2.2 Model for Concurrent Computation

For lack of space, and as we are dealing only with colorless tasks, we use a simplified version of the general combinatorial model of [9, 10, 11, 12, 13].

An *initial configuration* is an assignment of input values to processes, and a *final configuration* is the analogous for output values. A *task specification* consists of a set of legal initial configurations, and, for each of them, its corresponding set of legal final configurations.

A *colorless task* [12] is a triple $(\mathcal{I}, \mathcal{O}, \Delta)$, where \mathcal{I} is the *input complex*, \mathcal{O} is the *output complex*, and $\Delta : \mathcal{I} \rightarrow 2^{\mathcal{O}}$ is a carrier map. Each vertex in \mathcal{I} is an input value, and each simplex is an initial configuration, with possible initial configurations closed under inclusion. The output complex is analogous in regard to final configurations. Given an initial configuration, the carrier map Δ specifies which final configurations are legal.

With colorless tasks, only *sets* of input and output values define valid configurations, and such sets are closed under inclusion. Indeed, in k -set agreement [7], which is colorless, if a non-faulty process adopts the input of another non-faulty process, the initial configuration remains valid, and similarly for outputs. Conversely, in *renaming* [2], which is colored, the specific input/output assignments cannot be manipulated in such ways.

To illustrate the relation between the carrier map and the task specification, in binary consensus the input complex \mathcal{I} has two vertices, labeled 0 and 1, joined by an edge, and the output complex \mathcal{O} has two isolated vertices, labeled 0 and 1. The carrier map Δ sends each vertex of \mathcal{I} to the output vertex with the same label, and the edge of \mathcal{I} to both vertices of \mathcal{O} .

Non-faulty processes, informally speaking, start on vertices of a simplex σ in \mathcal{I} , and halt on vertices of a simplex $\tau \in \Delta(\sigma)$. Multiple processes may start on the same input vertex and halt on the same output vertex. Also, note that task specifications constrain the behavior of non-faulty processes

only. The same task definition is used both for crash and Byzantine failure models, with the nature of the failures affecting only *protocols* (algorithms), but not *specifications* (models). Moreover, the outputs of non-faulty processes depend solely on the inputs of non-faulty processes, which we indeed desire for colorless tasks subjected to Byzantine failures.

Operationally speaking, we have $n+1$ sequential processes³ that communicate via message-passing through a fully-connected, asynchronous network of reliable FIFO channels. Every message sent is eventually delivered after a finite, but unbounded delay, with the message’s sender reliably identified. The processes are asynchronous as well, having no bound on their relative speed. Up to t processes might be *faulty*, displaying arbitrary, “Byzantine” behavior. We abstract the asynchronous communication in discrete rounds of related messages, in a full-information protocol.

It follows from prior work [13]⁴ that in the asynchronous crash-failure model, a colorless task $(\mathcal{I}, \mathcal{O}, \Delta)$ has a t -resilient protocol if and only if:

Property 2.1. There is a continuous map $f : |\text{skel}^t(\mathcal{I})| \rightarrow |\mathcal{O}|$ carried by Δ .

The principal conclusion of this paper is to show that in the asynchronous Byzantine model, a task has a t -resilient protocol if and only if it satisfies Property 2.1, and, in addition, satisfies the following condition:

$$n + 1 > t(\dim(\mathcal{I}) + 2).$$

The Byzantine model places new constraints tying together $n+1$, the number of processes, $\dim(\mathcal{I})+1$, the maximum number of distinct input values in any initial configuration, and t , the maximum number of Byzantine processes. (In the crash failure model, these values are independent.)

3 Basic Algorithmic Tools

Our constructions make use of two well-known constructions from prior work: *reliable broadcast* [3, 5, 6, 18] and *stable vectors* [2].

3.1 Reliable Broadcast

Reliable broadcast forces Byzantine processes to communicate consistently with other processes. Communication is organized in asynchronous rounds,

³ Choosing $n+1$ processes rather than n simplifies the topological notation but slightly complicates the computing notation. Choosing n processes makes the opposite trade-off. We choose $n+1$ for compatibility with prior work.

⁴ The condition shown here is a simplified version of the original, reflecting our focus on colorless (instead of arbitrary) tasks.

where a round may involve several message exchanges. Messages have the form (P, r, c) , where P is the sending process, r is the current round, and c is the actual content. Messages not conforming to this structure can safely be discarded. Reliable broadcast gives the following guarantees [3, 6]:

Non-Faulty Integrity: If a non-faulty P never reliably broadcasts (P, r, c) , no non-faulty process ever reliably receives (P, r, c) .

Non-Faulty Liveness: If a non-faulty P does reliably broadcast (P, r, c) , all non-faulty processes will reliably receive (P, r, c) eventually.

Global Uniqueness: If two non-faulty processes Q and R reliably receive, respectively, (P, r, c) and (P, r, c') , then the messages are equal ($c = c'$), even if the sender P is Byzantine.

Global Liveness: For two non-faulty processes Q and R , if Q reliably receives (P, r, c) , then R will reliably receive (P, r, c) eventually, even if the sender P is Byzantine.

The protocol for reliable broadcast is discussed in detail in [3, 6], and works as long as $n + 1 > 3t$. For completeness, we give the algorithms in Appendix A. In this text, $P.\text{RBSend}(M)$ denotes the reliable broadcast of message M by process P , and $P.\text{RBRecv}(M)$ the reliable receipt of M by P .

3.2 Quorums

Let \mathcal{M}^r be the set of all messages reliably broadcast by all processes during a discrete round r , and let $M_i^r \subseteq \mathcal{M}^r$ be the subset reliably received by a non-faulty process P_i . By the global uniqueness of the reliable broadcast, if (P, r, c) and (P', r', c') are distinct messages in M_i^r then the senders are distinct: $P \neq P'$. Furthermore, by definition of M_i^r , we have that $r = r'$.

Let $\text{Good}(\mathcal{M}^r)$ denote the set of distinct message contents in \mathcal{M}^r that were reliably broadcast by the non-faulty processes. We say that a content c has a *quorum* in M_i^r , written $c \in \text{Quorum}(M_i^r)$, if c was reliably received in M_i^r from $t + 1$ or more different processes. If P has a quorum for c , P knows that c was sent by a non-faulty process. Conversely, any content received from less than $t + 1$ processes cannot be trusted – recall that non-faulty process outputs must depend only on non-faulty process inputs.

Algorithm 1 shows the procedure by which non-faulty processes get a set of messages containing a quorum. Informally, the procedure eventually finishes since all $n + 1 - t$ non-faulty processes eventually show up, and, since $n + 1 - t > t \cdot |\text{Good}(\mathcal{M}^r)|$, some value appears $t + 1$ or more times. The following lemmas states such guarantees, formally proven in Appendix B.

Algorithm 1 $P.\text{RecvQuorum}(r)$

```
 $M \leftarrow \emptyset$ 
while  $|M| < n - t + 1$  or  $\text{Quorum}(M) = \emptyset$  do
  upon  $\text{RRecv}((Q, r, c))$  do
     $M \leftarrow M \cup \{(Q, r, c)\}$ 

return  $M$ 
```

Property 3.1. If $n + 1 > 3t$ and $n + 1 > t(|\text{Good}(\mathcal{M}^r)| + 1)$, $\text{RecvQuorum}(r)$ eventually returns, and, for any non-faulty process P_i

$$|M_i^r| \geq n + 1 - t \text{ and } \text{Quorum}(M_i^r) \neq \emptyset.$$

Property 3.2. After executing $\text{RecvQuorum}(r)$, any two non-faulty processes P_i and P_j are such that $|M_i^r \setminus M_j^r| \leq t$.

3.3 Stable Vectors

Algorithm 1 ensures that any two non-faulty processes P_i and P_j will collect at least $n + 1 - 2t$ messages in common. Indeed, we have that $|M_i^r| \geq n + 1 - t$ by Property 3.1, and that $|M_i^r \setminus M_j^r| \leq t$ by Property 3.2, which implicates that $|M_i^r \cap M_j^r| \geq n + 1 - 2t$.

In Algorithm 2, we adapt the *stable vectors* technique presented in [2] to ensure that the two non-faulty processes P_i and P_j have $n + 1 - t$ messages in common in M_i^r and M_j^r , with a common value in $\text{Quorum}(M_i^r \cap M_j^r)$ and with sets of messages totally ordered by containment.

The technique works as follows. (1) P uses $\text{RecvQuorum}()$ to get a set of messages M with $\text{Quorum}(M) \neq \emptyset$. (2) P reliably transmits its *report*, containing those messages. (3) Any further message reliably received in M will make P reliably broadcast an updated report of M . (4) P keeps reliably receiving reports, stored in a vector R , until it identifies $n + 1 - t$ *buddies* in B , where a buddy is a process P_j whose last report R_j is M .

The following lemmas show that the procedure terminates and that the stable vector properties are indeed provided. Intuitively, any two non-faulty processes identify a common non-faulty buddy, which reliably broadcasts monotonically increasing reports, guaranteeing the properties. For formal proofs, refer to Appendix B. We assume that $n + 1 > 3t$ and also that $n + 1 > t(|\text{Good}(\mathcal{M}^r)| + 1)$.

Algorithm 2 $P.\text{RecvStable}(r)$

```

 $M, B \leftarrow \emptyset$ 
 $R[x] \leftarrow \emptyset$  for all  $0 \leq x \leq n$ 
 $M \leftarrow \text{RecvQuorum}(r)$ 
4:  $\text{RBSend}((P, r\{\text{report}\}, M))$ 
   while  $|B| < n + 1 - t$  do
     upon  $\text{RBRecv}((P_j, r, c))$  do
        $M \leftarrow M \cup \{(P_j, r, c)\}$ 
8:    $\text{RBSend}((P, r\{\text{report}\}, M))$ 

     upon  $\text{RBRecv}((P_j, r\{\text{report}\}, R_j))$  do
        $R[j] \leftarrow R_j$ 
12:  $B \leftarrow \{P_x : R[x] = M, 0 \leq x \leq n\}$ 
   return  $M$ 
```

Lemma 3.3. For any non-faulty process P_i , the sequence of transmitted reports is monotonically increasing. All other non-faulty processes receive those reports in such order.

Lemma 3.4. $\text{RecvStable}(r)$ eventually returns.

Lemma 3.5. After executing $\text{RecvStable}(r)$, any two non-faulty processes P_i and P_j have (i) $|M_i^r \cap M_j^r| \geq n + 1 - t$; (ii) $\text{Quorum}(M_i^r \cap M_j^r) \neq \emptyset$; and (iii) sets totally ordered by containment: $M_i^r \subseteq M_j^r$ or $M_j^r \subseteq M_i^r$.

4 k -Set Agreement and Barycentric Agreement

Our solvability theorem for colorless tasks builds on two novel asynchronous Byzantine protocols: k -set agreement [7], and *barycentric agreement* [13], which depend, in turn, on our previous communication primitives.

4.1 k -Set Agreement

Recall that in the k -set agreement task, the processes start on vertices of a simplex σ and halt on vertices of a simplex of $\text{skel}^{k-1}(\sigma)$.

Algorithm 3 shows a k -set agreement protocol assuming $\dim(\sigma) = d > 0$, $k > t$, and $n+1 > t(d+2)$. At most $d+1$ distinct values are reliably broadcast by non-faulty processes, so $|\text{Good}(\mathcal{M}^1)| = d+1$. As no more than t messages are missed by P , its decision is among the $(t+1)$ least-ranked elements, and

Algorithm 3 $P.KSetAgree(v)$

RBSend($(P, 1, v)$)
 $M \leftarrow \text{RecvQuorum}(1)$
return least-ranked element in $Quorum(M)$

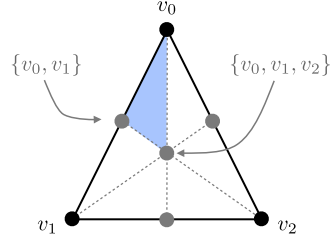


Figure 1: Barycentric subdivision of $\sigma = \{v_0, v_1, v_2\}$, and one of its simplices $\{\{v_0\}, \{v_0, v_1\}, \{v_0, v_1, v_2\}\}$.

non-faulty processes will certainly decide among those $(t + 1)$ least-ranked elements, for identical reason. The claim follows as $k > t$.

4.2 Barycentric Agreement

The barycentric subdivision of simplex σ is constructed, informally speaking, by subdividing σ along the barycenters of its faces, as shown in Figure 1. Formally, the barycentric subdivision of σ is a simplicial complex $\text{Bary } \sigma$ whose vertices are faces of σ , and whose simplices are chains of distinct faces totally ordered by containment. Every m -simplex $\tau \in \text{Bary } \sigma$ might be written as $\{\sigma_0, \dots, \sigma_m\}$, where $\sigma_0 \subset \dots \subset \sigma_m \subseteq \sigma$.

In the barycentric agreement task, non-faulty processes start on vertices of a simplex σ and halt on vertices of a simplex in $\text{Bary } \sigma$. Formally, for each $\sigma \in \mathcal{I}$, we have $\Delta(\sigma) = \text{Bary } \sigma$.

Algorithm 4 shows the barycentric agreement protocol, which assumes that $\dim(\sigma) = d > 0$ and $n + 1 > t(d + 2)$. By Lemma 3.5, for any two non-faulty processes P_i and P_j , we have that $M_i \subseteq M_j$ or $M_j \subseteq M_i$, and also that $Quorum(M_i \cap M_j) \neq \emptyset$. Therefore, $Quorum(M_i) \subseteq Quorum(M_j)$ or $Quorum(M_j) \subseteq Quorum(M_i)$, implicating that the decided values, which are faces of σ , are totally ordered by containment.

Algorithm 4 $P.BaryAgree(v)$

RBSend($(P, 1, \{v\})$)
 $M \leftarrow \text{RecvStable}(1)$
return $Quorum(M)$

5 The Solvability Theorem

In this section, we enunciate and prove our main theorem:

Theorem 5.1. A colorless task $(\mathcal{I}, \mathcal{O}, \Delta)$ has a t -resilient protocol in the asynchronous Byzantine model if and only if

1. $n + 1 > t(\dim(\mathcal{I}) + 2)$ and
2. there is a continuous map $f : |\text{skel}^t(\mathcal{I})| \rightarrow |\mathcal{O}|$ carried by Δ .

Proof. Conditions imply protocol. Say the map f exists. By the simplicial approximation theorem [16], we know that f has a *simplicial approximation* $\phi : \text{Bary}^N \text{skel}^t(\mathcal{I}) \rightarrow \mathcal{O}$, for some $N > 0$, also carried by Δ . The protocol for non-faulty processes is shown below, presuming $n + 1 > t(\dim(\mathcal{I}) + 2)$.

1. Call the Byzantine k -set agreement protocol, for $k = t + 1$, choosing vertices on a simplex in $\text{skel}^t(\mathcal{I})$.
2. Call the Byzantine barycentric agreement protocol N times to choose vertices in $\text{Bary}^N \text{skel}^t(\mathcal{I})$.
3. Use $\phi : \text{Bary}^N \text{skel}^t(\mathcal{I}) \rightarrow \mathcal{O}$ to choose vertices on a simplex in \mathcal{O} .

Since ϕ and f are carried by Δ , non-faulty processes starting on vertices of $\sigma \in \mathcal{I}$ finish on vertices of $\tau \in \Delta(\sigma)$. Also, since $\dim(\sigma) \leq \dim(\mathcal{I})$, the preconditions are satisfied for calling the protocols in steps (1) and (2).

Protocol implies conditions. Say the protocol exists. We argue by reduction to the crash-failure case. It is known [13] that in the crash failure model, if there is a t -resilient protocol, then there is a continuous map $f : |\text{skel}^t(\mathcal{I})| \rightarrow |\mathcal{O}|$ carried by Δ . But any t -resilient Byzantine protocol is also a t -resilient crash-failure protocol, so such a map exists even in the more demanding Byzantine model. The other condition is discussed below.

Number of processes. To show that $n + 1 > t(\dim(\mathcal{I}) + 2)$ is necessary in the Byzantine-failure model, we see how the $(t + 1)$ -set agreement task has no solution if $t + 2 \leq n + 1 \leq t(\dim(\mathcal{I}) + 2)$. Take an execution where Byzantine processes crash before taking any steps. If, for every $v \in \sigma \in \mathcal{I}$, t or fewer non-faulty processes have v as input, any non-faulty process P is unable to “trust” any other values, which makes $(t + 1)$ -set agreement impossible. \square

5.1 Non-Independent Faults

The t -resilient model presumes independent, identically distributed process failures, but in practice they correlate to the same processor, machine, etc.

The core/survivor-set model of Junqueira and Marzullo [14] assumes an adversarial scheduler that defines sets of possibly faulty/correct processes. Our algorithms and theorem **do apply** to this extended failure model – we give below some intuition on why they remain applicable.

In the reliable broadcast, quorum, and stable vector protocols, we redefine the *variable* t to be $c - 1$, one less than the minimum core size. Now, processes can always wait for $(n + 1) - (c - 1) = n + 1 - t$ messages, which preserves our correctness arguments for these protocols, still allowing us to solve c -set agreement and barycentric agreement. The solvability theorem for non-independent faults is stated below, and the trivial changes required for its demonstration are highlighted in Appendix C for completeness.

Theorem 5.2. A colorless task $(\mathcal{I}, \mathcal{O}, \Delta)$ has a protocol in the asynchronous Byzantine model with core size c if and only if $n + 1 > (c - 1) \cdot (\dim(\mathcal{I}) + 2)$ and there is a continuous map $f : |\text{skel}^{c-1}(\mathcal{I})| \rightarrow |\mathcal{O}|$ carried by Δ .

6 Conclusion

In this work, we give the first necessary and sufficient conditions for the solvability of arbitrary colorless tasks in asynchronous Byzantine systems. This particular characterization is interesting *per se*, as previous, analogous results exist for crash-failure systems [13]. However, this work also evidences (i) the power and suitability of the combinatorial topology tools, and (ii) novel, yet simple algorithms for the fundamental k -set agreement and barycentric agreement problems in asynchronous Byzantine systems.

We saw how combinatorial topology tools can produce novel results in distributed computing by virtue of being capable to support existential arguments without complicated, model-specific constructive arguments. Indeed, we used or slightly adapted basic communication primitives (reliable broadcast, quorums, stable vectors) and straightforward algorithmic tools (k -set agreement and barycentric agreement) to demonstrate our solvability theorem, relying on powerful, model-independent observations from combinatorial topology to complement the proof.

Our necessary and sufficient conditions capture the relation between the number of processes, the number of failures, and the topological structure of the task’s simplicial complexes. Hence, the solvability of asynchronous

Byzantine tasks *depend* on the ratio between benign and faulty processes (or core size), taking into account the input and output complexes. This differs from the case of crash-failure systems, and opens similar questions for other scenarios, such as synchronous systems and colored tasks, which we intend to carry as future work.

A Algorithms for Reliable Broadcast

Algorithms 5 to 7 show the reliable broadcast protocol for sender P , round r , and content c . The symbol “.” represents a wildcard, matching any value.

Algorithm 5 $P.\text{RBSend}((P, r, c))$

send(P, r, c) to all processes

Algorithm 6 $P.\text{RBEcho}()$

upon **recv**(Q, r, c) from Q **do**
 if never sent ($Q, r\{\text{echo}\}, \cdot$) **then**
 send($Q, r\{\text{echo}\}, c$) to all processes

upon **recv**($\cdot, r\{\text{echo}\}, c$) from $\geq n - t + 1$ processes **do**
 if never sent ($P, r\{\text{ready}\}, \cdot$) **then**
 send($P, r\{\text{ready}\}, c$) to all processes

upon **recv**($\cdot, r\{\text{ready}\}, c$) from $\geq t + 1$ processes **do**
 if never sent ($P, r\{\text{ready}\}, \cdot$) **then**
 send($P, r\{\text{ready}\}, c$) to all processes

Algorithm 7 $P.\text{RBRecv}((P, r, c))$

recv($\cdot, r\{\text{ready}\}, c$) from $n - t + 1$ processes
return (P, r, c)

B Proofs for Communication Primitives

In this section, we prove lemmas related to quorum and stable vectors. The proofs on stable vector guarantees are variations from those in [2].

B.1 Proof of Property 3.1

Proof. Since $n+1 > 3t$, the processes can perform reliable broadcast. Notice that the $n+1-t$ messages sent by the non-faulty processes can be grouped by their contents:

$$n - t + 1 = \sum_{c \in \text{Good}(\mathcal{M}^r)} |(P, r, c) : \{(P, r, c) \in \mathcal{M}^r, P \text{ is non-faulty}\}|.$$

If every content c in $\text{Good}(\mathcal{M}^r)$ was reliably broadcast by at most t non-faulty processes, we would have $n+1-t \leq |\text{Good}(\mathcal{M}^r)| \cdot t$, which contradicts the hypothesis. Hence, at least one content in $\text{Good}(\mathcal{M}^r)$ was reliably broadcast by more than $t+1$ non-faulty processes. By the non-faulty liveness of the reliable broadcast, such content will eventually be reliably received by all non-faulty processes. \square

B.2 Proof of Property 3.2

Proof. If $|M_i^r \setminus M_j^r| > t$, then M_j^r missed more than t messages in \mathcal{M}^r , the messages reliably broadcast in round r . However, this contradicts the fact that $|M_j^r| \geq n+1-t$ with M_j^r being obtained by reliable broadcast. \square

B.3 Proof of Lemma 3.3

Proof. First, note that the set M of P_i is monotonically increasing, so the sequence of transmitted reports is also monotonically increasing. As we assume FIFO channels, any other non-faulty process P_j receives those reports in the same order. \square

B.4 Proof of Lemma 3.4

Proof. Consider \mathcal{S} , the set of all messages reliably received by at least one non-faulty process. By the non-faulty liveness of the reliable broadcast, all non-faulty processes will eventually obtain $M = \mathcal{S}$, which is never expanded, or we would contradict the definition of \mathcal{S} . All non-faulty processes then send reports $\mathcal{R} = \mathcal{S}$, which are final, for the same reason as before.

Again, by the non-faulty liveness property of the reliable broadcast, all these processes will eventually receive $n + 1 - t$ reports \mathcal{R} . By the monotonicity of received reports (Lemma 3.3) and FIFO message delivery, those reports are not overwritten, matching the local $M = \mathcal{S}$. The non-faulty processes then return from `RecvStable()`. \square

B.5 Proof of Lemma 3.5

Proof. Call \mathcal{P}_i^r the set of processes whose reports are stored in R for process P_i and round r . Since all reports are transmitted via reliable broadcast, and every non-faulty process collects $n + 1 - t$ reports, $|\mathcal{P}_i^r \setminus \mathcal{P}_j^r| \leq t$ with $|\mathcal{P}_i^r| \geq n + 1 - t$, which implies that $|\mathcal{P}_i^r \cap \mathcal{P}_j^r| \geq n + 1 - 2t$. In other words, any two non-faulty processes identify $n + 1 - 2t > t + 1$ buddies in common, including a non-faulty process P_k . Therefore, $M_i^r = R'_k$ and $M_j^r = R''_k$, where R'_k and R''_k are reports sent by P_k at possibly different occasions.

Since the set M_k^r is monotonically increasing, either $R'_k \subseteq R''_k$ or $R''_k \subseteq R'_k$, guaranteeing property (iii). Both R'_k and R''_k contain R_k , the first report sent by P_k , by Lemma 3.3. Since $|R_k| \geq n + 1 - t$ and $Quorum(R_k) \neq \emptyset$, by Lemma 3.1, we guarantee properties (i) and (ii). \square

C Solvability for Non-Independent Faults

We here overview the changes in the proof of our solvability theorem between its principal formulation, and in the one following the model of Junqueira and Marzullo [14]. The overall proof structure is identical to the one for Theorem 5.1, so we only highlight some details, which are trivial.

Proof. Conditions imply protocol. Assuming the map f exists, proceed as before, but use the c -set agreement protocol to jump to $\text{skel}^{c-1}(\mathcal{I})$. The application of the barycentric agreement protocol and of the simplicial approximation theorem remain identical.

Protocol implies conditions. Assuming the protocol exists, we again reduce to the crash-failure case. It is known [12] that in the crash failure model, if there is a protocol with core size c , then we will have a continuous map $f : |\text{skel}^{c-1}(\mathcal{I})| \rightarrow |\mathcal{O}|$ carried by Δ , which suffices for our reduction.

Number of processes. Since we have defined $c = t + 1$, the argument holds verbatim. \square

References

- [1] I. Abraham, Y. Amit, and D. Dolev. Optimal resilience asynchronous approximate agreement. In *Proceedings of the 8th international conference on Principles of Distributed Systems*, OPODIS'04, pages 229–239, Berlin, Heidelberg, 2005. Springer-Verlag.
- [2] H. Attiya, A. Bar-Noy, D. Dolev, D. Peleg, and R. Reischuk. Renaming in an Asynchronous Environment. *Journal of the ACM*, July 1990.
- [3] H. Attiya and J. Welch. *Distributed Computing Fundamentals, Simulations, and Advanced Topics Second Edition*. John Wiley and Sons, Inc., 2004.
- [4] E. Borowsky and E. Gafni. Generalized FLP impossibility result for t -resilient asynchronous computations. In *STOC '93: Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, pages 91–100, New York, NY, USA, 1993. ACM.
- [5] G. Bracha. Asynchronous byzantine agreement protocols. *Information and Computation*, 75(2), 1987.
- [6] C. Cachin, R. Guerraoui, and L. Rodrigues. *Introduction to Reliable and Secure Distributed Programming*. Springer, 2nd ed. 2011 edition, Feb. 2011.
- [7] S. Chaudhuri. More choices allow more faults: Set consensus problems in totally asynchronous systems. *Information and Computation*, 105(1):132–158, July 1993.
- [8] M. Fischer, N. A. Lynch, and M. S. Paterson. Impossibility Of Distributed Commit With One Faulty Process. *Journal of the ACM*, 32(2), Apr. 1985.
- [9] M. Herlihy and S. Rajsbaum. Algebraic spans. *Mathematical Structures in Computer Science*, 10(4):549–573, 2000.
- [10] M. Herlihy and S. Rajsbaum. A classification of wait-free loop agreement tasks. *Theor. Comput. Sci.*, 291(1):55–77, 2003.
- [11] M. Herlihy and S. Rajsbaum. The topology of shared-memory adversaries. In *Proceeding of the 29th ACM SIGACT-SIGOPS symposium on Principles of distributed computing*, PODC '10, pages 105–113, New York, NY, USA, 2010. ACM.

- [12] M. Herlihy and S. Rajsbaum. Simulations and reductions for colorless tasks. In *Proceedings of the 2012 ACM symposium on Principles of distributed computing*, PODC '12, pages 253–260, New York, NY, USA, 2012. ACM.
- [13] M. Herlihy and N. Shavit. The topological structure of asynchronous computability. *J. ACM*, 46(6):858–923, 1999.
- [14] F. P. Junqueira and K. Marzullo. Designing Algorithms for Dependent Process Failures. Technical report, 2003.
- [15] D. N. Kozlov. *Combinatorial Algebraic Topology*, volume 21 of *Algorithms and Computation in Mathematics*. Springer, 1 edition, Oct. 2007.
- [16] J. Munkres. *Elements of Algebraic Topology*. Prentice Hall, 2 edition, Jan. 1984.
- [17] M. Saks and F. Zaharoglou. Wait-Free k-Set Agreement is Impossible: The Topology of Public Knowledge. *SIAM J. Comput.*, 29(5):1449–1483, 2000.
- [18] T. Srikanth and S. Toueg. Simulating authenticated broadcasts to derive simple fault-tolerant algorithms. *Distributed Computing*, 2, 1987.